I temporarily prefer to go with the coding option. Here are the two project ideas that I proposed below:

Project Idea1: Security Camera

Description:

The project idea is to design a security camera for detecting any “Intruder” that attempts to open a personal computer without the authorization of the owner.

Essential Functions:

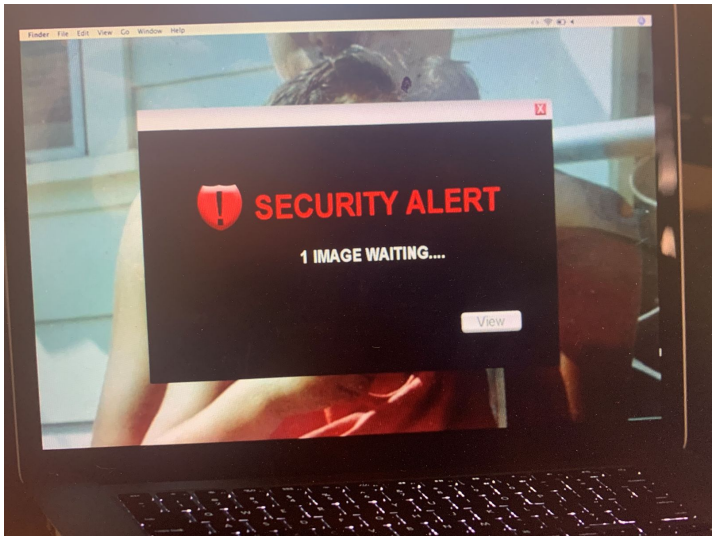
- Being able to capture the profile of “Intruder”
- Send a “Security Alert email” to owner immediately after the Intruder is being detected
- Don’t show the “Security Alert notification” until owner successfully login to his/her PC
- The Intruder shouldn’t know his picture was taken when he attempted to access PC

Extra functions

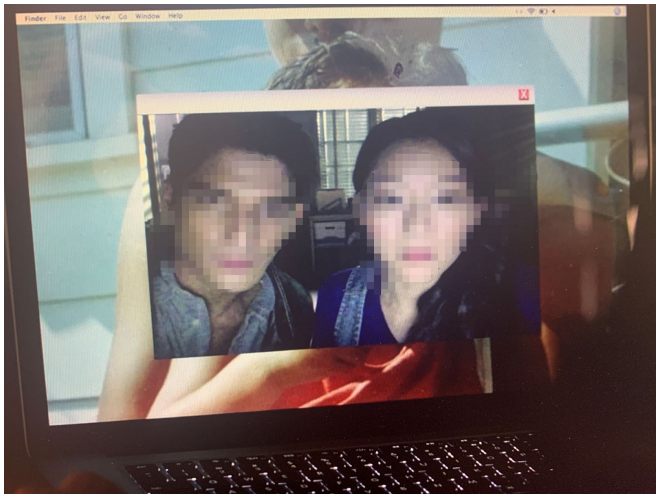
- Securly protect all the data store in PC if the “Intruder” attempt to plugin a portable flash card, e.g. NO copy/detect authorization allowed until I confirm this operation via email.

Expected result:

- After owner successfully login to his/her account, if “Intruder” being detected:



- A similar image should be pop up, after clicked “View”:



Project Idea2: Adding Security on Web Application

Description:

Based on the current web application that I developed on CSE3901, which can be found in my github, https://github.com/drago1234/Full-Stack__Project06-Photo-Sharing, to build a stronger security protection for the app. It should have a more robust security protection for user-login portal and provide an extra security protection for user databases. Specific security features includes but not limit to:

1. Check for unauthorized access:
    a. Say, if you have a page with a list of posts that a user owns, one link might be to /post/1. However, the user can easily go to a different post page by changing 1 to any number, e.g. /post/50.
2. Authentication: Save the hash of the password instead of the actual password. When authenticating a user, get the hash of the password and compare it with the hash on the database. Suggesting gems such as, device, authlogic, bcrypt, and salt.
3. Filter Passwords and other sensitive data on logs
    a. Rails logs all requests to your application. When a user logs in, the username and password will be logged unless you filter the password. Rails by default creates config/initializers/filter_parameter_logging.rb which has:

```
Rails.application.config.filter_parameters += [:password]
```

    b. We're not saving the passwords in clear text on the database and we don't want them to appear on the logs either. You'll want to filter other sensitive data like credit cards.

4. Cross Site Request Forgery (CSRF)
    a. Adding CSRF protection when your user submits a POST action. This is provided by Rails which by adding an authenticity token on forms.
5. Throttling Requests
    a. Prevent Rack attack, which is like trying thousands of passwords simultaneously.
6. Use HTTPS for the whole site
7. No Credentials in Repository: T
    a. he secret key base, database credentials, and other sensitive data should not be committed to your repository.
8. Design a better password policy to prevent password crack, which includes rules like:
    a. Case sensitivity
    b. At least 8 characters
    c. requires numeric, special character
    d. Not part of email or name
9. Prevent the SQL Injection exploits
10. Adding Bundler Audit: a bundle feature to check if any of the gems you're using has a vulnerability issue.

Options things to consider:

- Cross-site scripting
- Command injection
- SQL injection
- File access
- Mass assignment
- Unsafe code evaluation / code injection
- Disabled cross-site forgery protection
- Unsafe deserialization
- Open redirects
- Hard-coded secrets in source code
- Unsafe metaprogramming
- Session manipulation
- Unsafe session settings
- Unscoped database queries
- Weak hashing algorithms
- Skipped SSL certificate verification
- Dynamic render locations
- Unquoted HTML attributes
- Exposed error information
- Denial of service via dynamic regular expressions
- Basic Authentication with hard-coded passwords
- Missing httponly flag on cookies
- Rails-related CVEs

Reference:

1. RUBY ON RAILS SECURITY 17-ITEM CHECKLIST, https://blog.engineyard.com/ruby-on-rails-security-checklist
2. Rails SQL Injection, https://rails-sqli.org/
3. Railsconf 2015: The World of Rails Security, https://www.youtube.com/watch?v=AFOlxqQCTxs
4. Official Rails Security Guide, http://guides.rubyonrails.org/security.html
5. OWASP Rails Cheatsheet, https://www.owasp.org/index.php/Ruby_on_Rails_Cheatsheet
6. Secure Headers Gem, https://github.com/twitter/secureheaders#readme
7. Brakeman - Static Analysis Security Tool for Rails, http://brakemanscanner.org/